

Neural Network Based Energy Storage System Modeling for Hybrid Electric Vehicles

S.R. Bhatikar, R.L. Mahajan, K. Wipke, and
V. Johnson



NREL

National Renewable Energy Laboratory

1617 Cole Boulevard
Golden, Colorado 80401-3393

NREL is a U.S. Department of Energy Laboratory
Operated by Midwest Research Institute • Battelle • Bechtel

Contract No. DE-AC36-98-GO10337

Neural Network Based Energy Storage System Modeling for Hybrid Electric Vehicles

S.R. Bhatikar, R.L. Mahajan, K. Wipke, and
V. Johnson

Prepared under Task No. HV916010



able Energy Laboratory

1617 Cole Boulevard
Golden, Colorado 80401-3393

NREL is a U.S. Department of Energy Laboratory
Operated by Midwest Research Institute • Battelle • Bechtel

Contract No. DE-AC36-98-GO10337

NOTICE

This report was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or any agency thereof.

Available to DOE and DOE contractors from:
Office of Scientific and Technical Information (OSTI)
P.O. Box 62
Oak Ridge, TN 37831
Prices available by calling 423-576-8401

Available to the public from:
National Technical Information Service (NTIS)
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
703-605-6000 or 800-553-6847
or
DOE Information Bridge
<http://www.doe.gov/bridge/home.html>



Printed on paper containing at least 50% wastepaper, including 20% postconsumer waste

Abstract

The modeling of the energy storage system (ESS) of a hybrid electric vehicle (HEV) poses a considerable challenge. The problem is not amenable to physical modeling without simplifying assumptions that compromise the accuracy of such models. An alternative is to build conventional empirical models. Such models however, are time-consuming to build and are data-intensive.

In this paper, we demonstrate the application of an artificial neural network (ANN) to modeling the ESS. The model maps the system's state-of-charge (SOC) and the vehicle's power requirement to the bus voltage and current. We show that neural network models can accurately capture the complex, non-linear correlations accurately. Further, we propose and deploy our new technique, Smart Select, for designing neural network training data. The underlying principle of Smart Select is to design training data such that it is uniformly distributed over the entire range of an appropriate ANN output variable, which is typically the variable that is most difficult to model. In this case, we selected training data that was uniformly distributed over the current range. We show that smart-select is economical in comparison with conventional techniques for selection of training data. Using this technique and our in-house neural network software (the CUANN), we developed an artificial neural network model (inputs=2, hidden neurons=3, outputs=2) utilizing only 4047 of the available 29,244 points. When validated on the remaining points, its predictive accuracy, measured by R-squared error, was 0.97.

Finally, we describe the integration of the ESS neural network model into the MATLAB-SIMULINK environment of NREL's Advanced Vehicle Simulator (ADVISOR).

Table of Contents

1.0 Introduction.....	1
2.0 The Artificial Neural Network.....	2
3.0 Energy Storage System (ESS)	4
4.0 Neural Network Modeling of the ESS	7
4.1 Dynamic Neural Networks	8
4.2 Static Neural Networks.....	12
4.3 The Smart-Select Technique	13
4.4 ANN with Smart-Select.....	14
4.5 Sensitivity to Temperature.....	14
5.0 ANN integration into ADVISOR.....	16
5.1 Investigation of Unsatisfactory Performance.....	20
6.0 ANN with Hawker Data	22
7.0 Conclusions and Results	24
8.0 Potential Areas for Investigation.....	26
9.0 Acknowledgements	26
Appendix A: The original empirical-analytical implementation of the ESS module in ADVISOR	A-i
Appendix B: The C++ program for Smart-Select.....	B-i
Appendix C: Porting between CUANN and MATLAB	C-i

List of Figures

Figure 1: The HEV system is comprised of subsystems.....	1
Figure 2: The artificial neuron	2
Figure 3: The non-linear sigmoid activation function.....	2
Figure 4: An artificial neural network.....	3
Figure 5: A schematic of the ESS.....	4
Figure 6: Principle of the ESS simulation	5
Figure 7: Original ESS simulation in ADVISOR	5
Figure 8: Optima Data.....	6
Figure 9: Schematic of neural network models in ESS simulation, with Optima data	7
Figure 10: Schematic of a dynamic neural network models for ESS simulation	8
Figure 11: Dynamic ANN Performance – Voltage.....	10
Figure 12: Dynamic ANN Performance – Current	11
Figure 13: Static ANN Performance – Current	12
Figure 14: Uneven distribution of data vis-B-vis current	13
Figure 15: Histogram showing even distribution of data vis-B-vis current.....	14
Figure 16: ANN Performance with Smart-Select	15
Figure 17: Integration of ANN in ADVISOR	17
Figure 18: Initial Failed ANN Integration in ADVISOR.....	18
Figure 19: Target Results (Original Algorithm)	19
Figure 20: ANN Prediction HWFET	20

List of Figures (concluded)

Figure 21: Offset in Experimental A-H Data	21
Figure 22: ESS Formulation with Regression	22
Figure 23: Uneven distribution of raw data vis-B-vis current.....	23
Figure 24: Even distribution of training data vis-B-vis current with Smart-Select.....	23
Figure 25: ANN Validation Performance.....	24
Figure 26: ANN Prediction on HWFET.....	24
Figure 27: Successful ANN integration ADVISOR	25

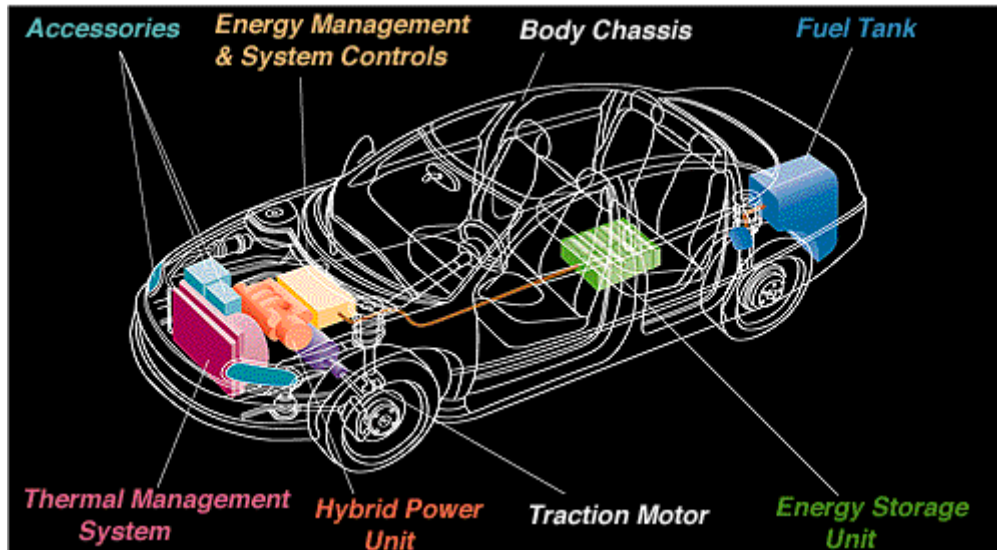
List of Tables

<u>Table I:</u> Description of OPTIMA data files.....	27
<u>Table II:</u> Range of the input and output variables for the driving cycles.....	27
<u>Table III:</u> Range of the variable 'Power Requested' for ADVISOR driving cycles ..	28

1.0 Introduction

A growing dependence on foreign oil, along with a heightened concern over the environmental impact of personal transportation, has led the U.S. government to investigate and sponsor research into advanced transportation concepts. One of these future technologies is the hybrid electric vehicle (HEV), typically featuring both an internal combustion engine and an electric motor, with the goal of producing lower emissions while obtaining superior fuel economy. Figure 1 below lists the typical components found in an HEV.

The Department of Energy's National Renewable Energy Laboratory (NREL) has developed an HEV simulator, called the ADvanced VehIcle SimulatOR (ADVISOR). This simulator facilitates the optimization of HEV configurations with different subsystems, for best fuel economy and emission level. ADVISOR requires models of the individual components of a HEV, such as the propulsion unit and the energy storage unit.



One way to develop models of HEV components is through rigorous analytical procedure. This is typically time-consuming and the simplifying assumptions required to make the analysis tractable impair the value of such models. An alternative is to employ conventional empirical methods, which are variations of the classical regression theme. These models are unwieldy and are usually only suitable for low-end non-linearities. In this paper, we present the artificial neural network as a practical alternative to analytical and empirical methods that is accurate and easy to use.

2.0 The Artificial Neural Network

Neural networks are computational models of the biological brain. The biological brain of an adult human is composed of several billion neurons. If all the neurons in one adult human brain were laid out end-to-end, they would stretch for several hundred miles. While each neuron is functionally simple, the neurons are massively interconnected, through adjustable, directed links. It is believed that this parallel distributed processing architecture of the human brain is responsible for its remarkable abilities.

A neural network is comprised of artificial neurons. An artificial neuron, like its biological counterpart, is a simple computational element. It first performs a weighted sum of its inputs (see Figure 2). This sum is referred to as the *activation* of the neuron. Then, the neuron applies a non-linear *sigmoid* transformation (see Figure 3) to modulate its activation.

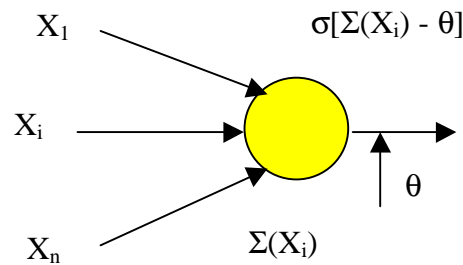


Figure 2: The artificial neuron.

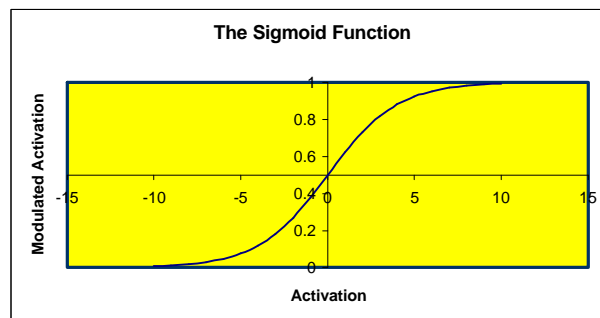


Figure 3: The non-linear sigmoid activation function.

The neural network is composed of a layered arrangement of neurons that are interconnected (see Figure 4) by weighted interconnections. Note the three-layered structure of the schematic representation in Figure 4. The INPUT layer has a neuron for

each input, the OUTPUT layer has a neuron for each output, and the HIDDEN layer (there may be several hidden layers) comprises processing neurons. These interconnection weights (represented by matrices $[W_1]$ and $[W_2]$) are adjustable.

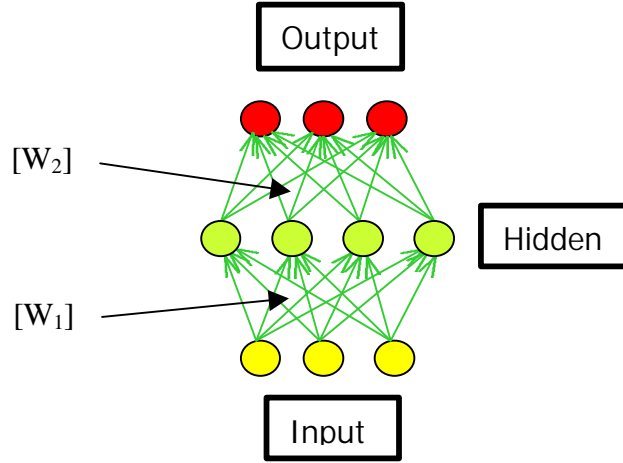


Figure 4: An artificial neural network.

Such a neural network translates into a mathematical function. For the network with one hidden layer, as shown in Figure 3, the function is

$$\underline{Y} = F\left(\sum_{\text{hidden}} \underline{W}^{(2)} \times F\left(\sum_{\text{inputs}} \underline{W}^{(1)} \times \underline{I}\right)\right)$$

Where:

- \underline{I} is the vector of inputs;
- \underline{W} is the matrix of interconnection weights, with its superscript denoting the layer as per Figure 3 ($\underline{W}_1 = \underline{W}^{(1)}$);
- \underline{Y} is the vector of outputs.

The power of artificial neural networks lies in the theorem which says that given sufficient hidden neurons the function represented by an artificial neural network can approximate any function, however non-linear, to arbitrary accuracy in a finite domain. A neural network starts out with random weights, and the weights are adjusted until the required degree of accuracy is obtained. In the context of a neural network, this is *learning*. To train a neural network an algorithm called backpropagation is employed.

With backpropagation, the convergence of a neural network to the mapping underlying its training data is guaranteed.

3.0 Energy Storage System (ESS)

Batteries are used for storage of electrical energy in hybrid electric vehicles. Batteries store and deliver electrical energy chemically by initiating and reversing chemical reactions respectively. One ESS option is the conventional lead-acid battery. Lead-acid technology is mature and economical. In addition to the lead-acid battery, there are newer options such as nickel metal hydride (Ni-MH) and lithium-ion (Li-ion) batteries. The ESS is normally comprised of a bank of batteries in series.

Of all the sub-systems constituting a hybrid electric vehicle, the energy storage system is probably the most difficult to understand and model. Although a battery is a simple electrical energy storage device that delivers and accepts energy, the highly non-linear nature of its electrochemical processes makes it difficult to model.

Figure 5 represents the general scheme of the ESS module as required to be implemented in ADVISOR. This block accepts a power request (P_r), and depending on the state-of-charge (SOC) of the battery pack, returns the bus voltage (V) and current (I). The ESS output power is simply the product of the bus voltage and current.



Figure 5: A schematic of the ESS.

In the original version of ADVSIOR, the ESS model was based on the circuit as shown in Figure 6. The open-circuit voltage of the battery-pack (V_{oc}) and its internal resistance (R_{int}) are computed as functions of the SOC. Applying straightforward circuit law analysis, I and V are computed from V_{oc} , R_{int} and P_r .

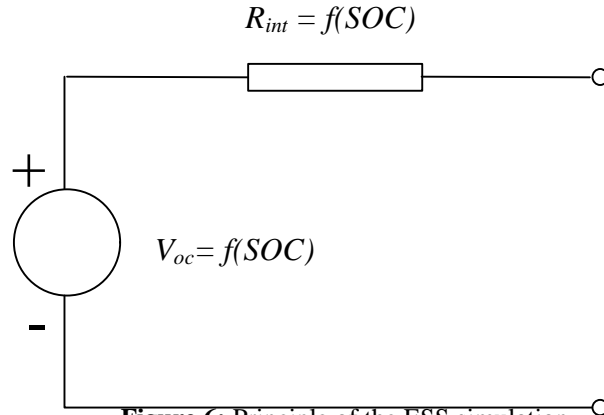


Figure 6: Principle of the ESS simulation.

Implementation details of the ESS simulation in the MATLAB-SIMULINK graphical programming environment are shown in Figure 7. Figure 7 comprises four internal modules:

- a module to compute pack voltage and internal resistance;
- a module to limit output power;
- a module to compute bus voltage and current;
- a module to update the battery SOC.

Step-by-step explanations of each internal module are given are provided in Appendix A.

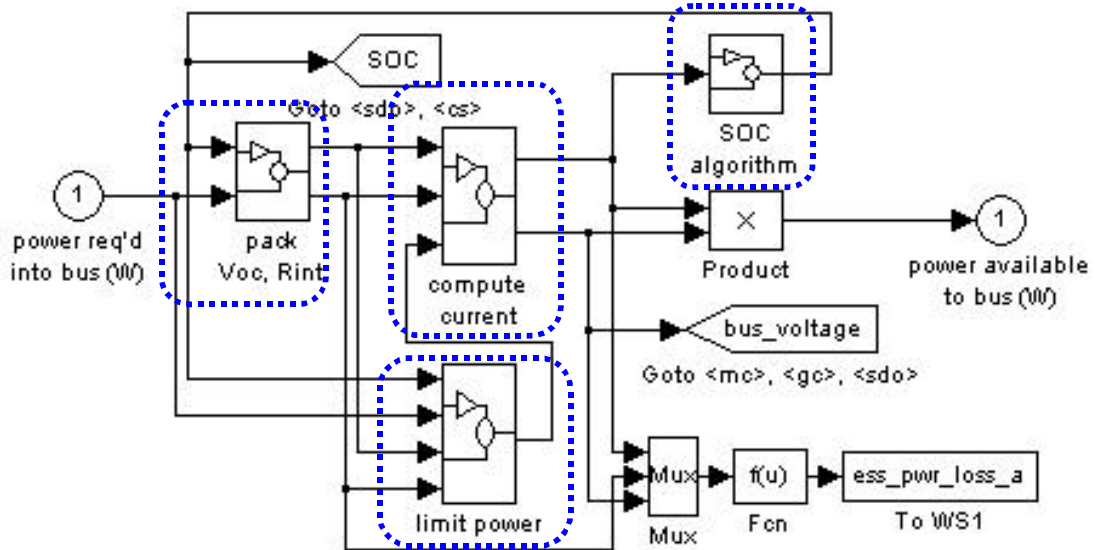
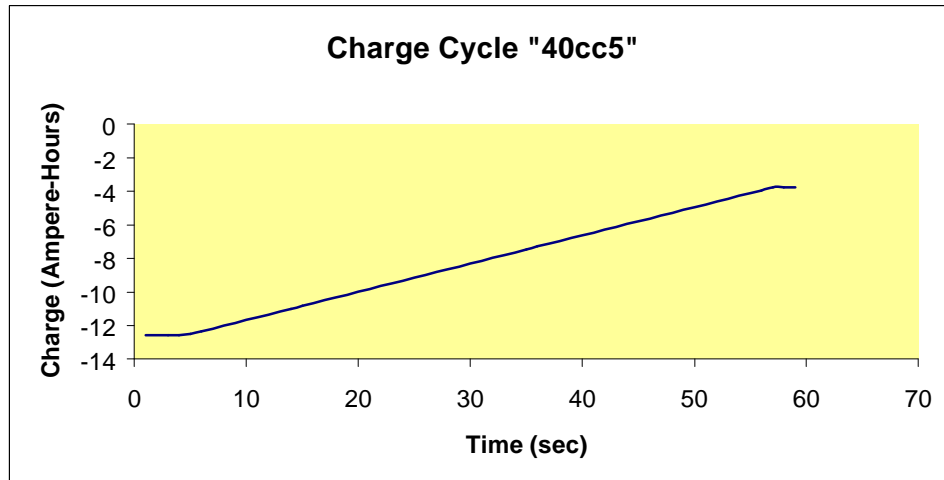
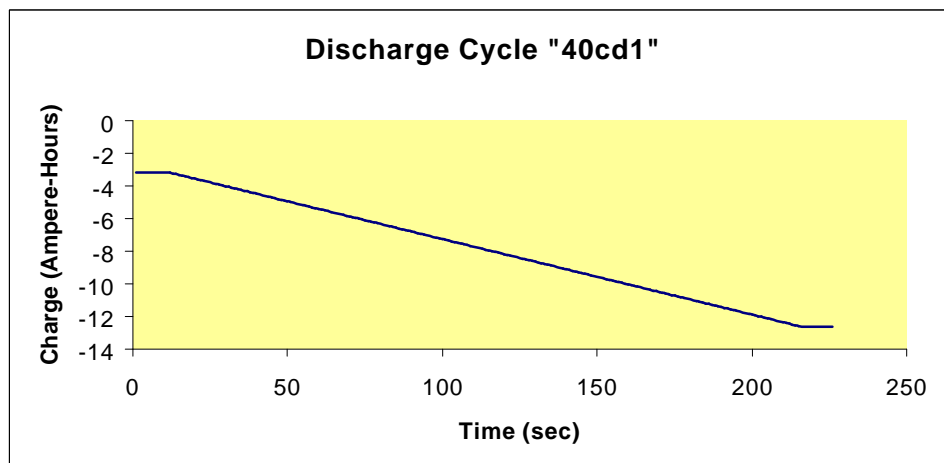


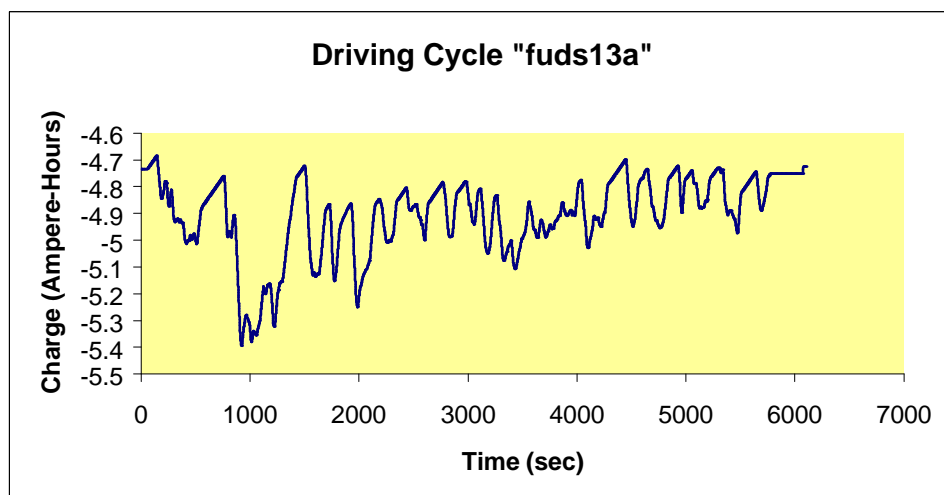
Figure 7: Original ESS simulation in ADVISOR.



A. Charge Cycle "40cc5"



B. Discharge Cycle "40cd1"



C. Driving Cycle "fuds13a"

Figure 8: Optima Data

This simulation is based on empirical correlations between the SOC and the open-circuit voltage (V_{oc}) and internal resistance (R_{int}) of the battery pack. The bus voltage and current are derived from P_r , V_{oc} and R_{int} analytically, by application of circuit law. The empirical correlations are piece-wise regression models. Develop of the regression models is a time-consuming task. It would be simpler to collect battery data and develop a neural network to predict bus voltage and current with P_r and SOC as its inputs. Accordingly, battery data was collected and neural network based models of the ESS were developed. The neural networks mapped SOC and P_r (inputs) to V and I (outputs).

4.0 Neural Network Modeling of the ESS

To start with, neural networks were trained with data collected from an Optima lead-acid battery for charge, discharge and driving cycles. One example of each type of cycle is shown in Figure 8. The files are described in Table 1. The schematic of the neural network is shown in Figure 9.

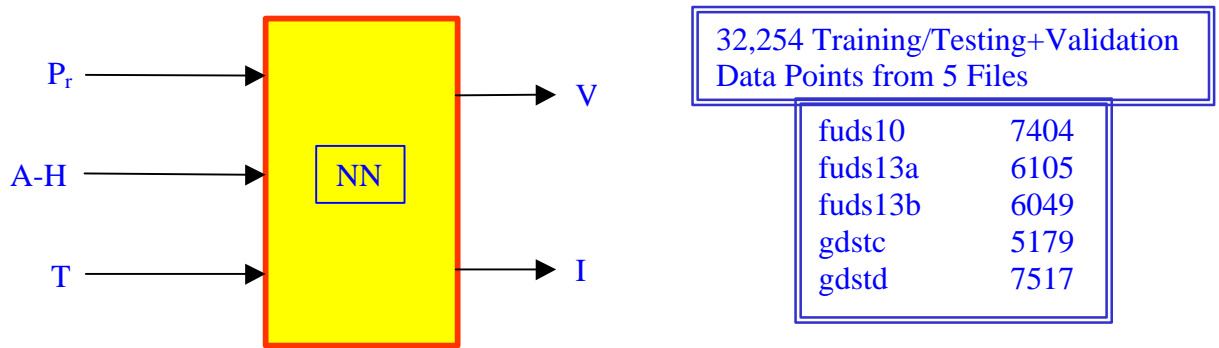


Figure 9: Schematic of neural network models in ESS simulation, with Optima data.

Notice that the average battery temperature is an input to the network. Temperature was included as an input because it influences the electrochemical processes of a battery. Also note that the state of charge of the battery pack was represented by its non-normalized value in Ampere-Hours (A-H). Further, it was reasoned that it would be best to use only the data from the driving cycles for training neural networks. This is because driving cycle data is representative of the operative performance of a battery, and is therefore more likely to capture process dynamics than a charge or discharge cycle. Thus, neural

networks were developed with a total of 32,254 data points for training, testing and validation.

4.1 *Dynamic Neural Networks*

In order to capture the dynamic element of battery behavior, a dynamic neural network was constructed. The schematic of a dynamic neural network is shown in Figure 10.

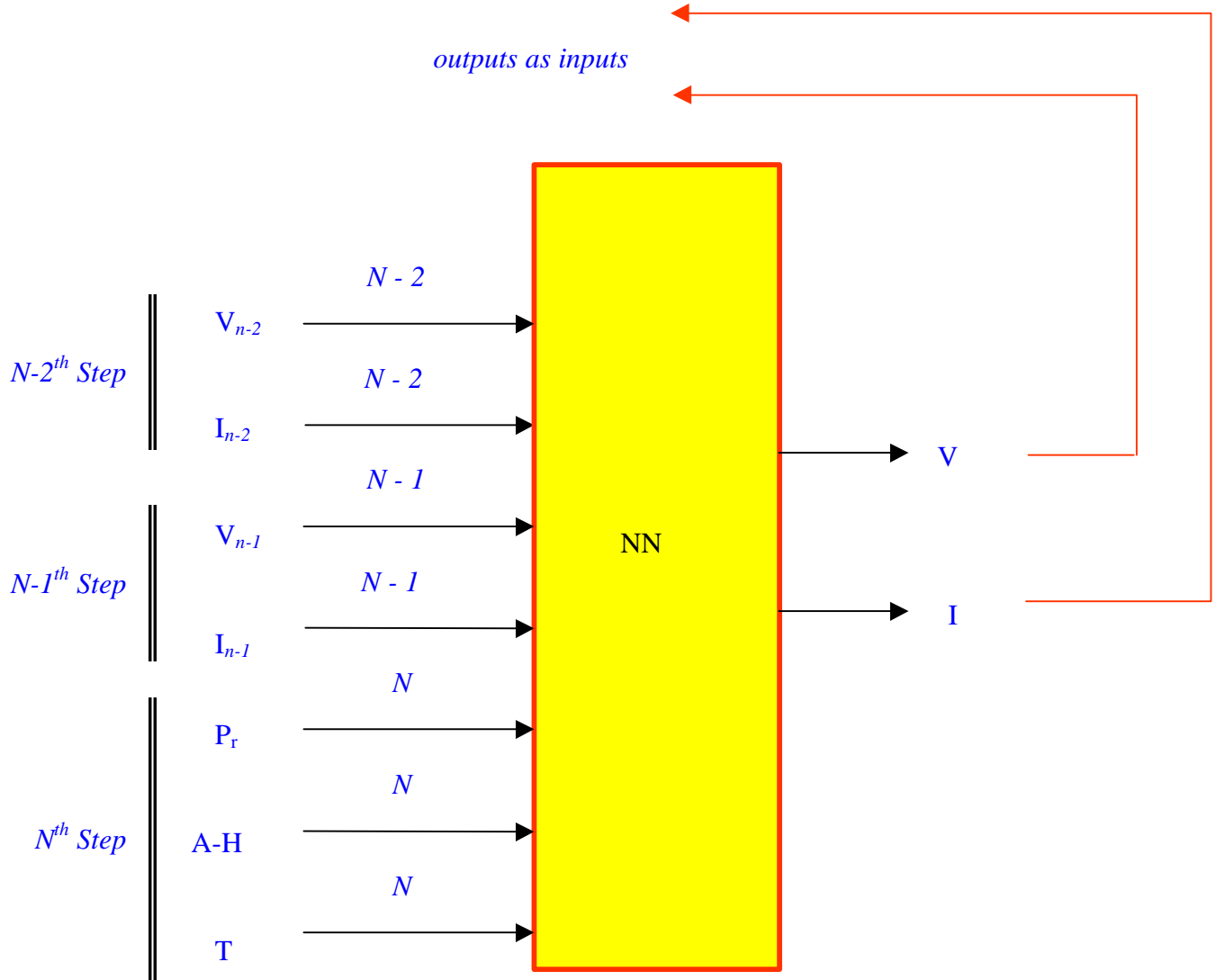


Figure 10: Schematic of a dynamic neural network models for ESS simulation.

In a dynamic neural network, one or more of the outputs is used as input with time delay. This means, that the selected output variables at one step serve as inputs for the next step.

Multiple time delay elements may be employed, as shown in Figure 10. Such networks capture the dynamics of the process, i.e. the influence of previous states on the subsequent states. Neural networks were built with one, two and three dynamic elements for both outputs (V and I).

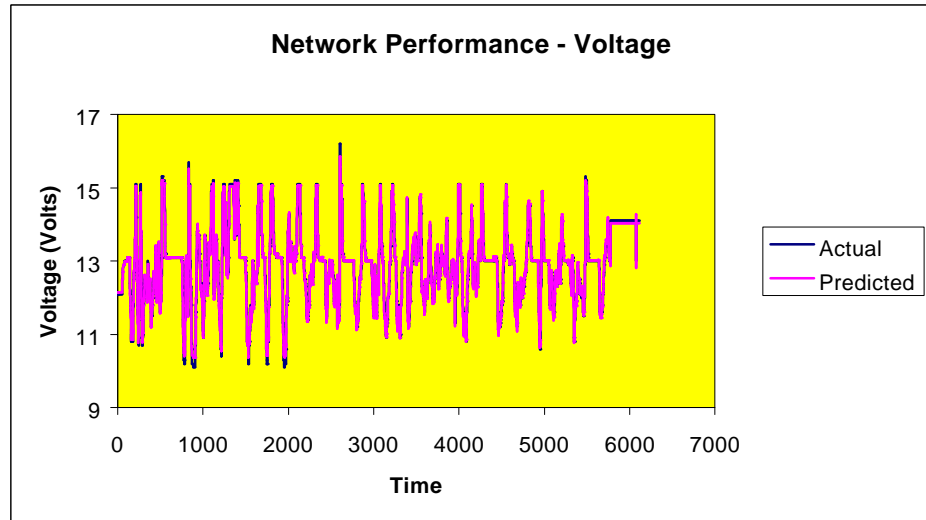
It was observed that the dynamic neural networks did not perform very well. Further, the performance of a neural network with one time delay was the best and that of the network with three time delays was the worst.

Figures 11 and 12 show the results of training a neural network with one time delay. The driving cycle “fuds13a” was employed for training, testing and validation. Of the 6105 data points of this cycle, 2399 points, evenly distributed in chronological sequence, were used for training and testing, and the network was validated upon all the 6105 points.

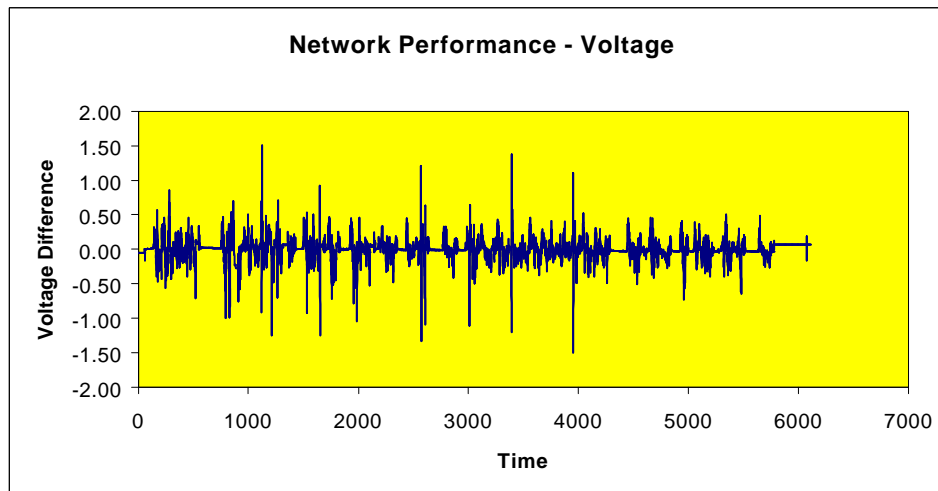
Figure 11 shows the neural network performance on voltage prediction and Figure 12 shows the performance on current prediction. Figure 11-(A) shows the predicted and actual voltage on the time axis. The prediction error is shown in Figure 11-(B). Figure 11-(C) is a plot of the actual voltage versus the predicted voltage. The red line indicates 100% accuracy of prediction. Clearly, the network does a reasonable job of voltage prediction. The average prediction error¹ is 0.27% and the MSE is nearly zero. However, Figure 12 shows that the prediction of current can be improved further. The average prediction error is 1.47% and the mean squared error (MSE) is 0.12, while the R-squared error is 0.96.

The improvement of performance with fewer time delay elements strongly suggests that a static model could be more accurate. It is possible that the dynamic element of the model was captured in the SOC algorithm, used in conjunction with the neural network. It was therefore decided to develop static neural networks.

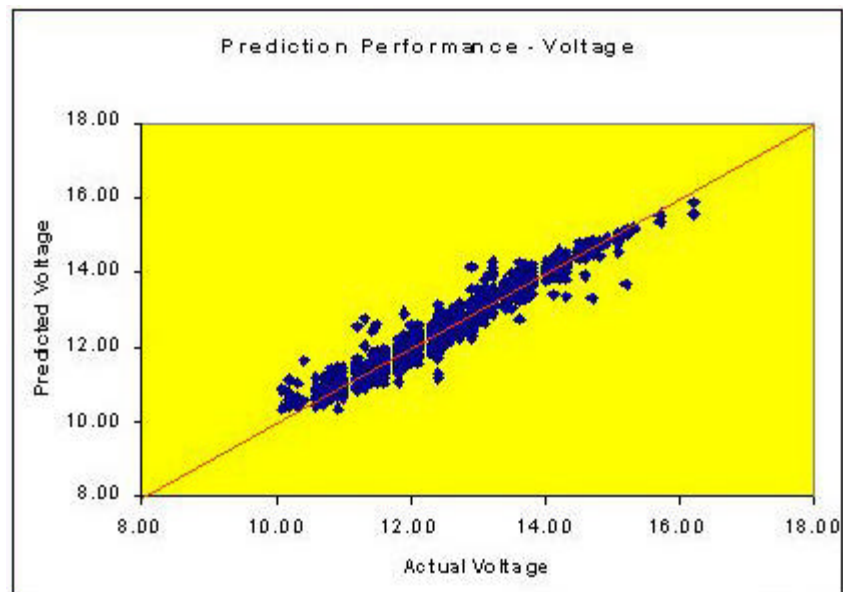
¹ The percent error was measured relative to the prediction range, as the ratio of the difference between the target and predicted values to the prediction range.



A.



B.



C.

Figure 11: Dynamic ANN Performance - Voltage

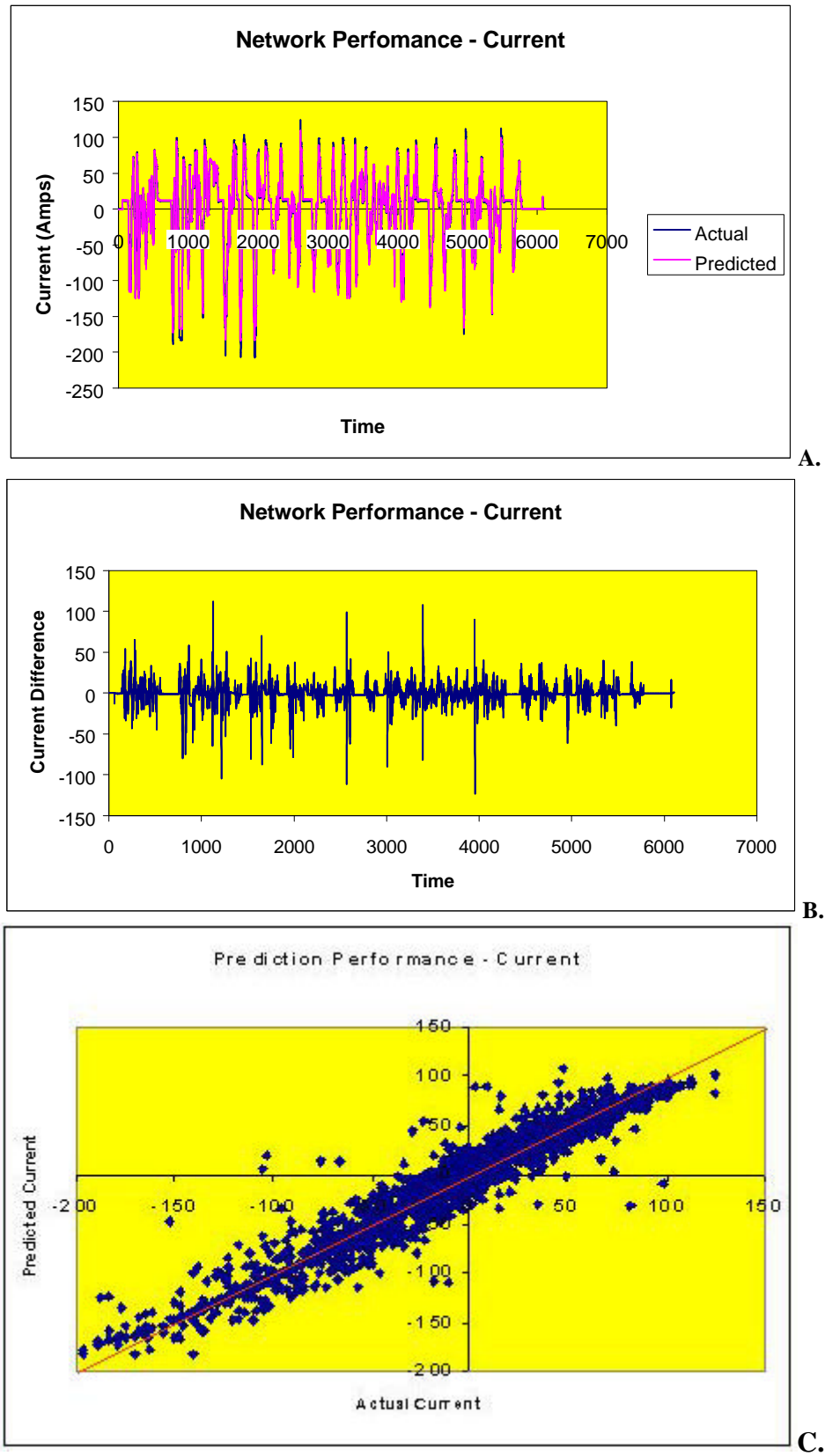


Figure 12: Dynamic ANN Performance – Current.

4.2 *Static Neural Networks*

To build static neural networks, training data was sampled at random from each driving cycle. The training data comprised 6000 data points (1200 from each driving cycle). Neural networks were trained and then validated on the entire data set of 32,254 points.

A clear improvement in performance was observed. Compare Figure 13 and 12-(C). Figure 13 is a plot of the actual current versus the predicted current, for validation over the driving cycle “fuds13a”. The error in prediction of current was 0.53%. The MSE is 0.08 and the R-squared error is 0.99. Observe, moreover, from Figure 13, that the error is severe only in the tail portion of the current space. In fact, this peculiar behavior was observed for all static networks trained with data randomly sampled from the driving cycles.

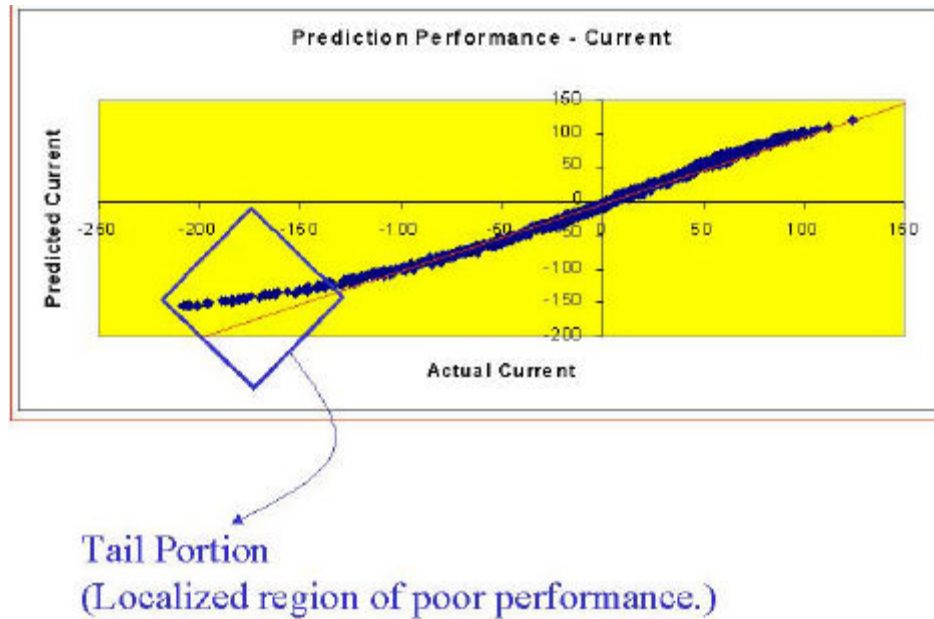


Figure 13: Static ANN Performance - Current.

It was determined that the distribution of training data was uneven over the current space. This is clearly reflected in Figure 14, which is a histogram of the data distribution vis-a-vis current, for the driving cycle “fuds13a”. More than 70% of the data is confined to less than 30% of the current space.

It was therefore decided to sample training data for an even distribution over the current space. This is our *smart-select* technique, described in the next section.

4.3 The Smart-Select Technique

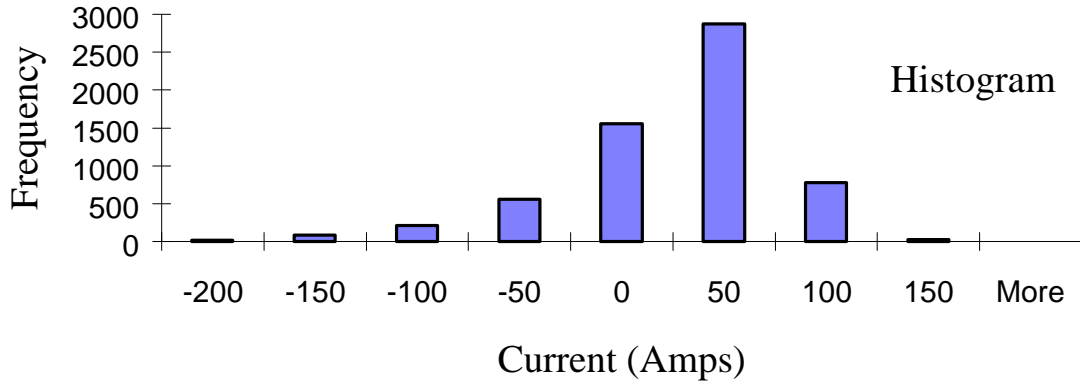


Figure 14: Uneven distribution of data vis-à-vis current.

The underlying principle of the Smart-Select technique is to design training data such that it is uniformly distributed over the entire range of an appropriate output variable. This variable is the one that is most difficult to model by the neural network. Conventional Design of Experiments (DOE) mandates an even distribution of training data over all variables – the inputs as well as the outputs. Such a full-factorial DOE scheme is data-intensive. It is subject to the ‘curse of dimensionality’ whereby the size of the training data set increases exponentially with the resolution of the variables. The problem of data-explosion is attenuated by a partial-factorial DOE. But, even with partial factorial DOE, the time required to select training data from an available data set is prohibitively large. This problem is what Smart-Select solves. Applying Smart-Select, the data is sampled so that the training data has an even distribution vis-à-vis one output variable only. The variable selected is the output variable that is more intractable than the others. For the neural network model of the ESS, the intractable output variable was the current. Accordingly, the training data was designed with the objective of ensuring an even distribution of data with respect to this variable. See Figure 15.

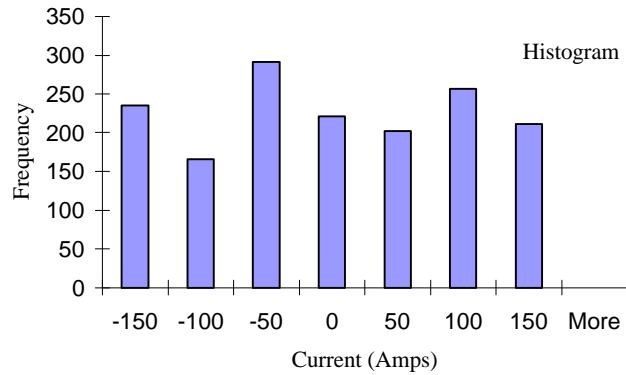


Figure 15: Histogram showing even distribution of data vis-à-vis current.

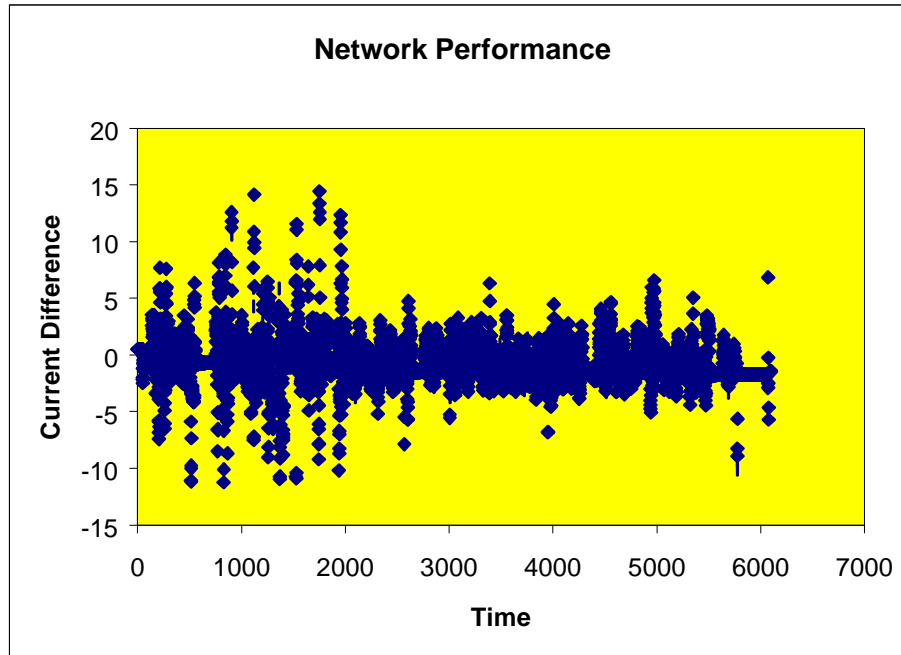
4.4 ANN with Smart-Select

The Smart-Select technique was implemented by a C++ program (see Appendix B). 1583 data points were sampled from the driving cycles. The points were sampled in approximately equal measure from each driving cycle.

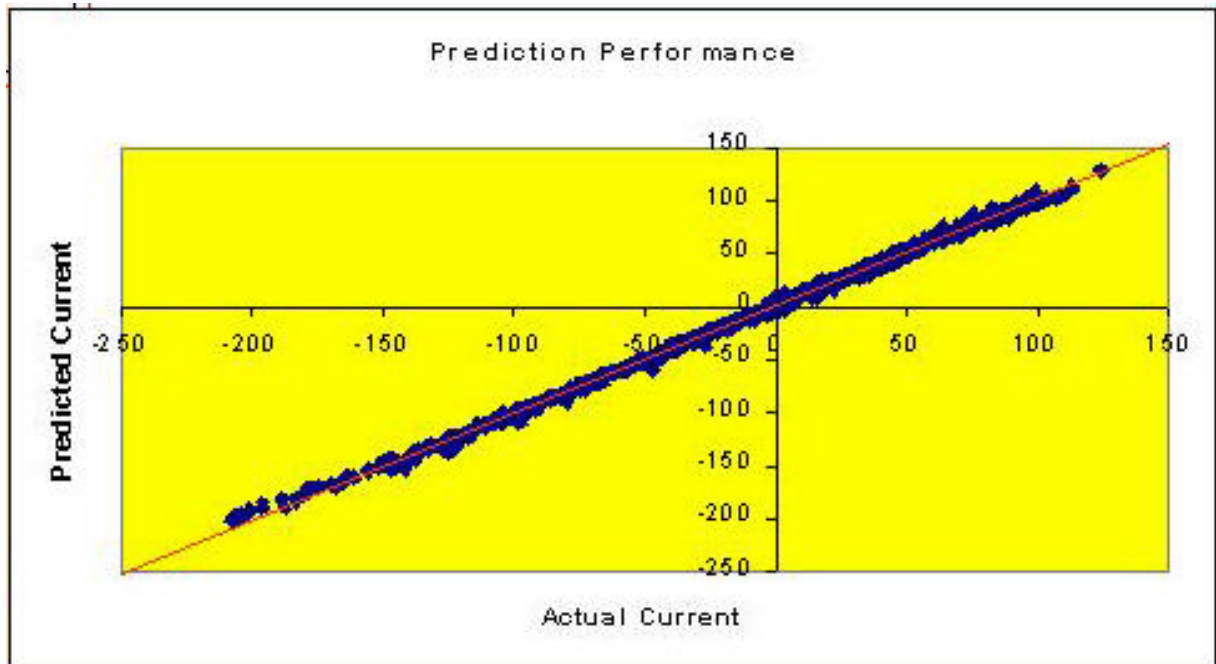
Neural networks were trained on these 1583 points. The best neural network had 1 hidden layer with 4 neurons. Figure 16 shows the performance of the neural network for validation over the driving cycle “fuds13a”. Figure 16-(A) is a plot of the prediction error, and Figure 16-(B) is a plot of the actual current versus the predicted current. The average prediction error is 0.6%, the MSE is 0.0057 and the R-squared error is 0.9993.

4.5 Sensitivity to Temperature

To test the sensitivity of the ESS model to temperature, temperature was dropped as an input and a neural network was trained as in 4.4, with the same training data but minus the temperature information. There was no deterioration in the network



A.



B.

Figure 16: ANN Performance with Smart-Select.

performance. Since it is known that temperature exerts significant influence on battery performance, it was determined that the effect of monotonically increasing temperature over the driving cycle was masked by the monotonically decreasing SOC. As temperature added no value to the input, it was excluded from further consideration.

5.0 ANN Integration into ADVISOR

At this stage, the static neural network described in section 4.5 was incorporated into ADVISOR. Neural networks were created in MATLAB, using the Neural Network Toolbox. [A back-propagation neural network is created in MATLAB using the “net” command. (See Appendix C.) The SIMULINK block diagram of a neural network is created using the “gensim” command.]

Note that pilot tests conducted on neural networks implemented using MATLAB showed inferior learning capability vis-a-vis the CUANN. Therefore, networks created in MATLAB were used only in the prediction mode, and network weights were derived from the CUANN. (See Appendix C for how the transfer of weights was performed.)

Accordingly, the network of 4.4, without the temperature input, was created. It was ascertained that this network functioned correctly in the prediction mode. Then, the network was incorporated into the SIMULINK block diagram of the ESS as shown in Figure 17.

The results of an ESS simulation with the ANN incorporated in ADVISOR are shown in Figure 18. Figure 19 shows the results of a matching ESS simulation with the original ADVISOR algorithm. A comparison of Figures 18 and 19 clearly reveals that this neural network implementation is unacceptable. As seen in Figure 18, the achieved vehicle speed falls consistently short of the required speed (topmost graph) for the duration of the driving cycle. This is unlike the results of the original ESS simulation for the same driving cycle in Figure 19, where the achieved speed matches the desired speed at all times (topmost graph). The shortfall in speed is a result of a lack of power. Since the original ESS simulation results indicate that the ESS is capable of meeting the power request throughout this driving cycle, the lack of power is a result of poor predictive capability of the neural network.

Figure 17: Integration of ANN in ADVISOR

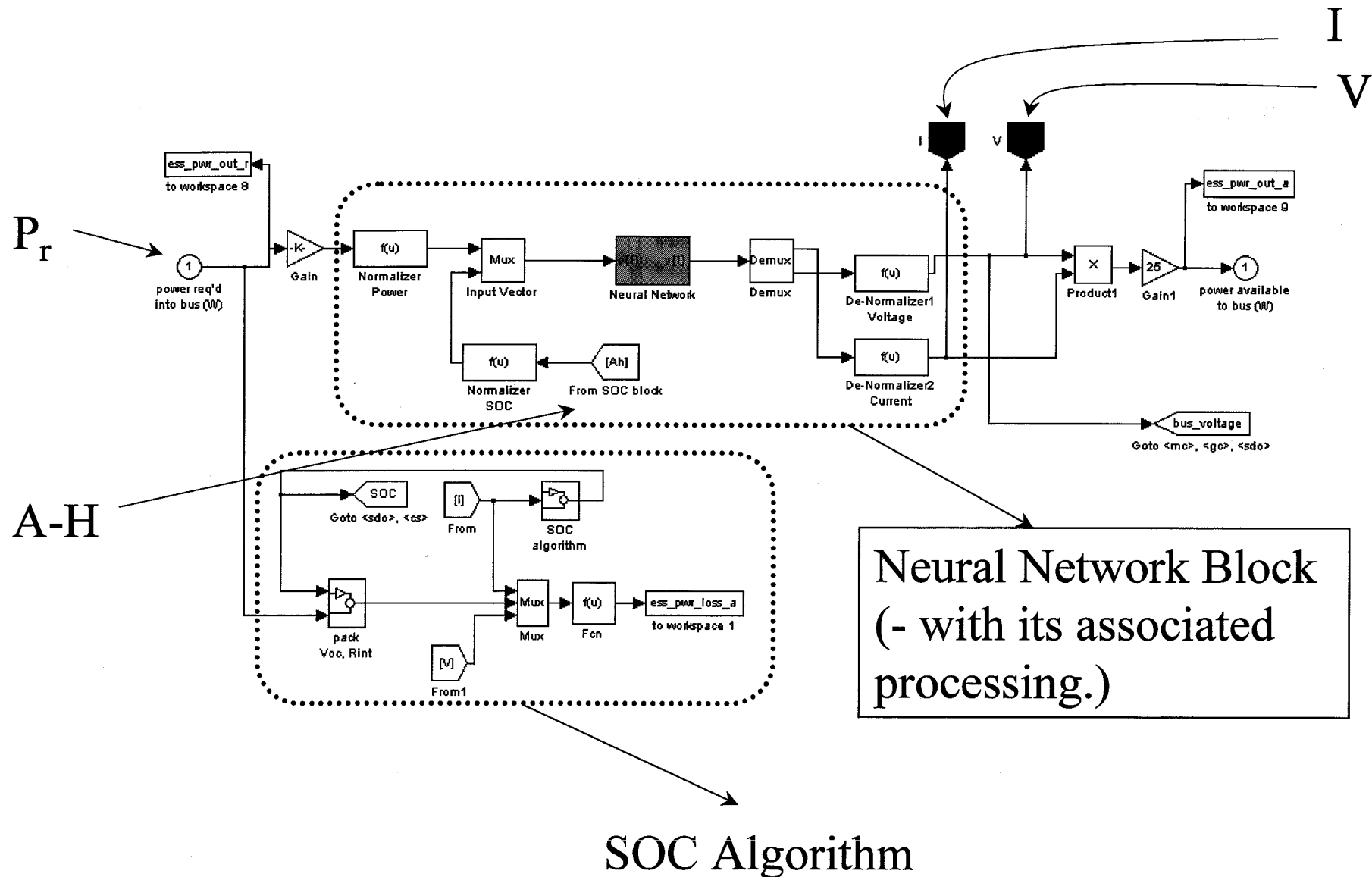


Figure 18: Initial Failed ANN Integration in ADVISOR

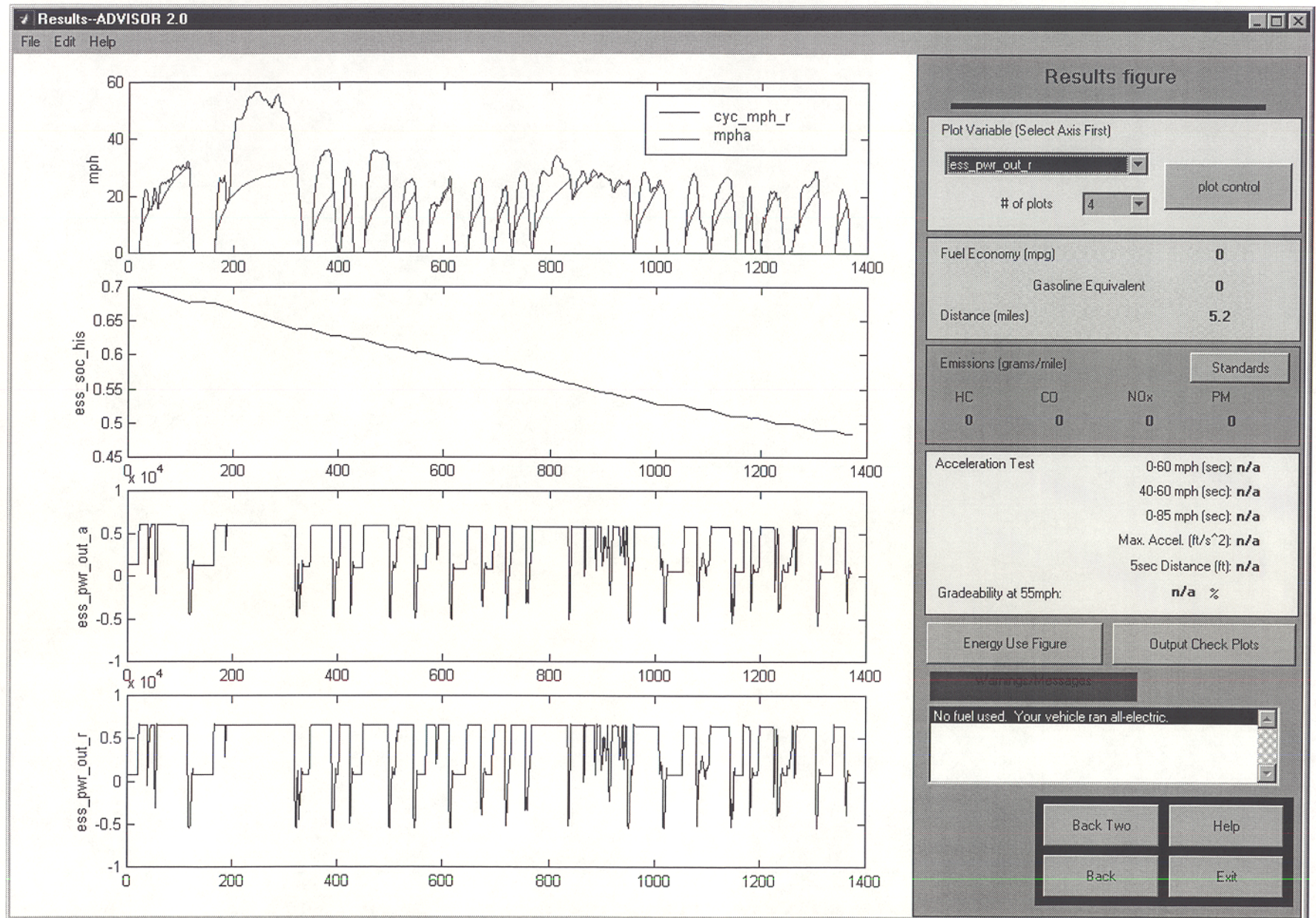
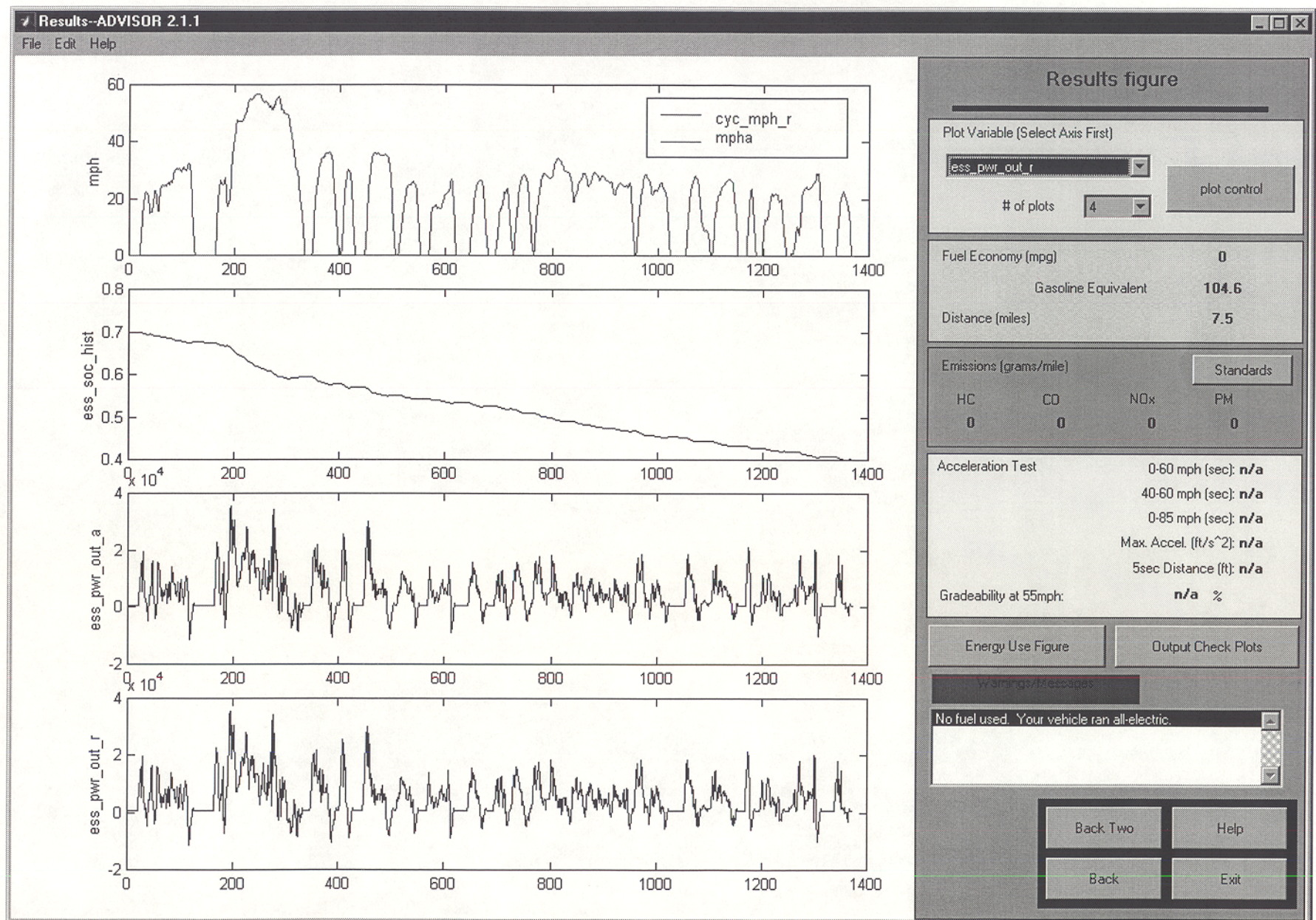


Figure 19: Target Results (Original Algorithm)



5.1 Investigation of Unsatisfactory Performance

Figure 20 shows the performance of the ANN on the driving cycle “HWFET” of ADVISOR. It is a plot of the power requested versus the power achieved over the HWFET driving cycle. The red line is representative of 100% prediction accuracy. The network is accurate only over a small region of the power space.

The explanation for this is straightforward: *poor prediction accuracy is observed where the prediction points are outside the range of the training data*. In other words, the network’s performance was poor for data outside the range of its training data. This is clearly reflected in Figure 20 - the range of the input variable P_r in the training data is marked with a thick line on the horizontal axis.

Table II shows the range of each of the input and output variables for the experimental data and the training data. From table III, it is seen that the range of the input variable P_r in the case of some of the ADVSIOR driving cycles, exceeds its range in the training data.

Further investigation revealed that the experimental A-H data was not being updated according to the SOC algorithm in ADVISOR. The SOC algorithm is explained

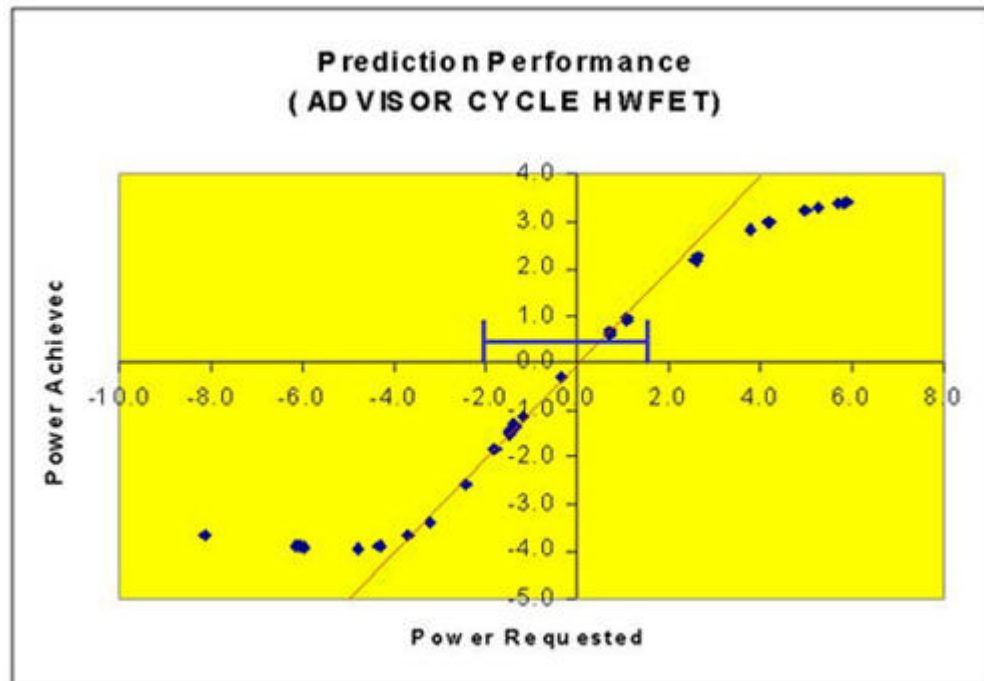


Figure 20: ANN Prediction on HWFET.

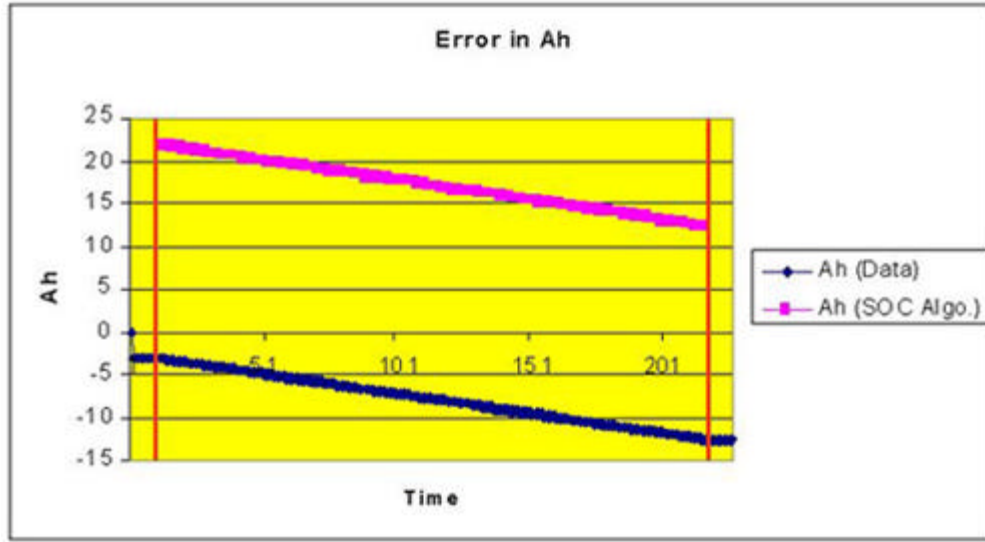


Figure 21: Offset in Experimental A-H Data.

in Appendix A (**Block 4: SOC Algorithm**). Figure 21 clearly shows that the A-H computed by the SOC algorithm is offset from the experimental A-H data by a constant amount for each data point of the cycle. The initial SOC was not properly initialized.

Since new driving cycle data had been collected for a Hawker lead-acid battery, which had more consistent tabulation of net Ah with SOC, the Optima data was replaced by the Hawker data. To verify the experimental data, the ESS model was formulated through regression models. The original ESS model employed look-up tables to correlate the SOC to the open-circuit voltage and internal resistance as explained in Appendix A. (**Block 1: Computation of Pack Open Circuit Voltage and Internal Resistance**). A regression analysis was performed and piece-wise regression models were developed to replace the look-up tables. (See Appendix III.) With these regression models, and circuit analysis laws (see Appendix A, **Block 3: Compute Current and Voltage**) a formulation was developed for the ESS, correlating the input variables (P_r and SOC) with the output variables (V and I). See Figure 22. The Hawker data was verified with respect to this formulation, to ensure its suitability for training the network.

6.0 ANN with Hawker Data

Neural networks were developed with the Hawker data. It was noted that the experimental data acquisition rig introduced noise due to inconsistent sampling rate of multiple data acquisition systems.

$$Ah \equiv SOC \rightarrow \text{Regression Analysis} \rightarrow \begin{matrix} V_{oc} \\ R_{int} \end{matrix}$$

$$I = \frac{V_{oc} - \sqrt{V_{oc}^2 - 4R_{int}P_r}}{2R_{int}}$$

$$V = V_{oc} - IR_{int}$$

Figure 22: ESS Formulation with Regression.

The new data set comprised 29,244 points of one driving cycle. These points were employed for training, testing and validation. Figure 23 is a histogram showing the distribution of the raw data vis-à-vis current. The non-uniformity of distribution of data vis-à-vis current is evident. Therefore, as before, the Smart-Select methodology was applied for selection of training data. 4047 points, sampled by Smart-Select, were employed for training and testing. Figure 24 is a histogram of the training data vis-à-vis current. The entire set of 29,244 points of the driving cycle was employed for validation. The best neural network had one hidden layer with three neurons. Figure 25 shows the validation performance of the neural network on a set of 2000 points of the driving cycle. The network performance is adequate. The average prediction error on the entire Hawker data set is 1.16%. The MSE is 0.0247 and the R-squared error is 0.9652. These results are not the best obtained so far. However, they are comparable to the results of prediction with the ESS formulation on the same data, which yields an average prediction error of 0.99, an MSE of 0.02 and an R-squared error of 0.973. Figure 26 shows, the performance

of the network on the driving cycle HWFET of ADVISOR. The average prediction error is 0.0108. The MSE is 0.15. The R-squared error is 0.988. As Figure 27 shows the neural network's performance now meets the requirements of the ADVISOR simulation.

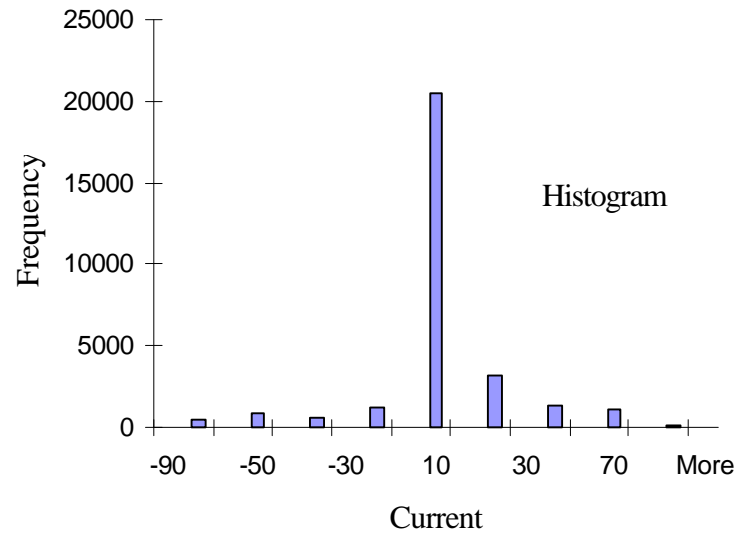


Figure 23: Uneven distribution of raw data vis-à-vis current.

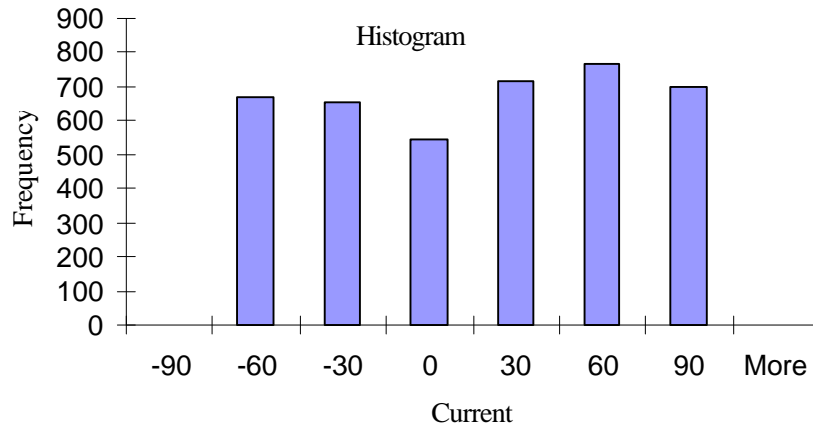


Figure 24: Even distribution of training data vis-à-vis current with Smart-Select.

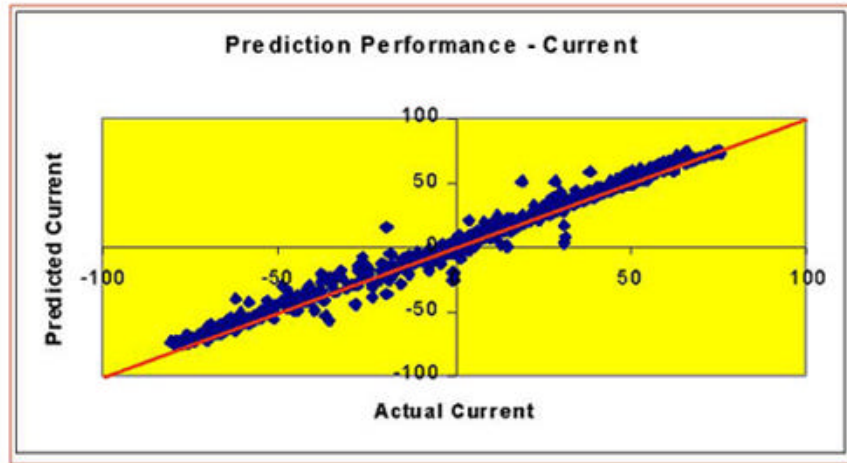


Figure 25: ANN Validation Performance.

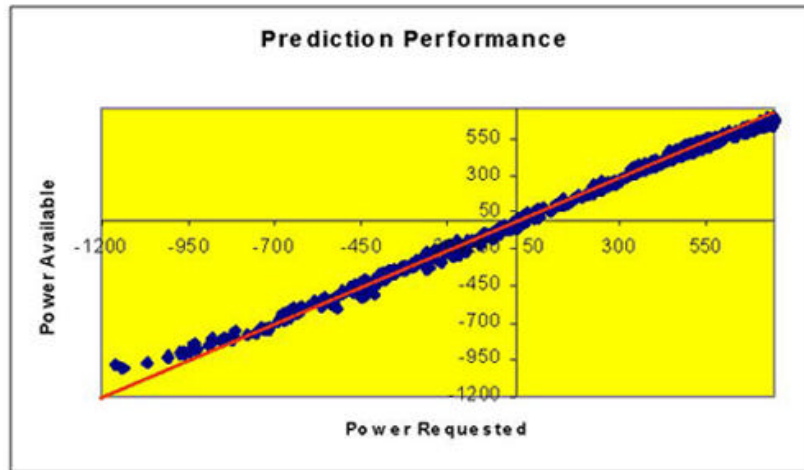


Figure 26: ANN Prediction on HWFET.

7.0 Conclusions and Results

We have demonstrated that the ESS of an HEV can be adequately modeled by the artificial neural network. We have also demonstrated the effectiveness of the Smart-Select technique for design of training data for an ANN.

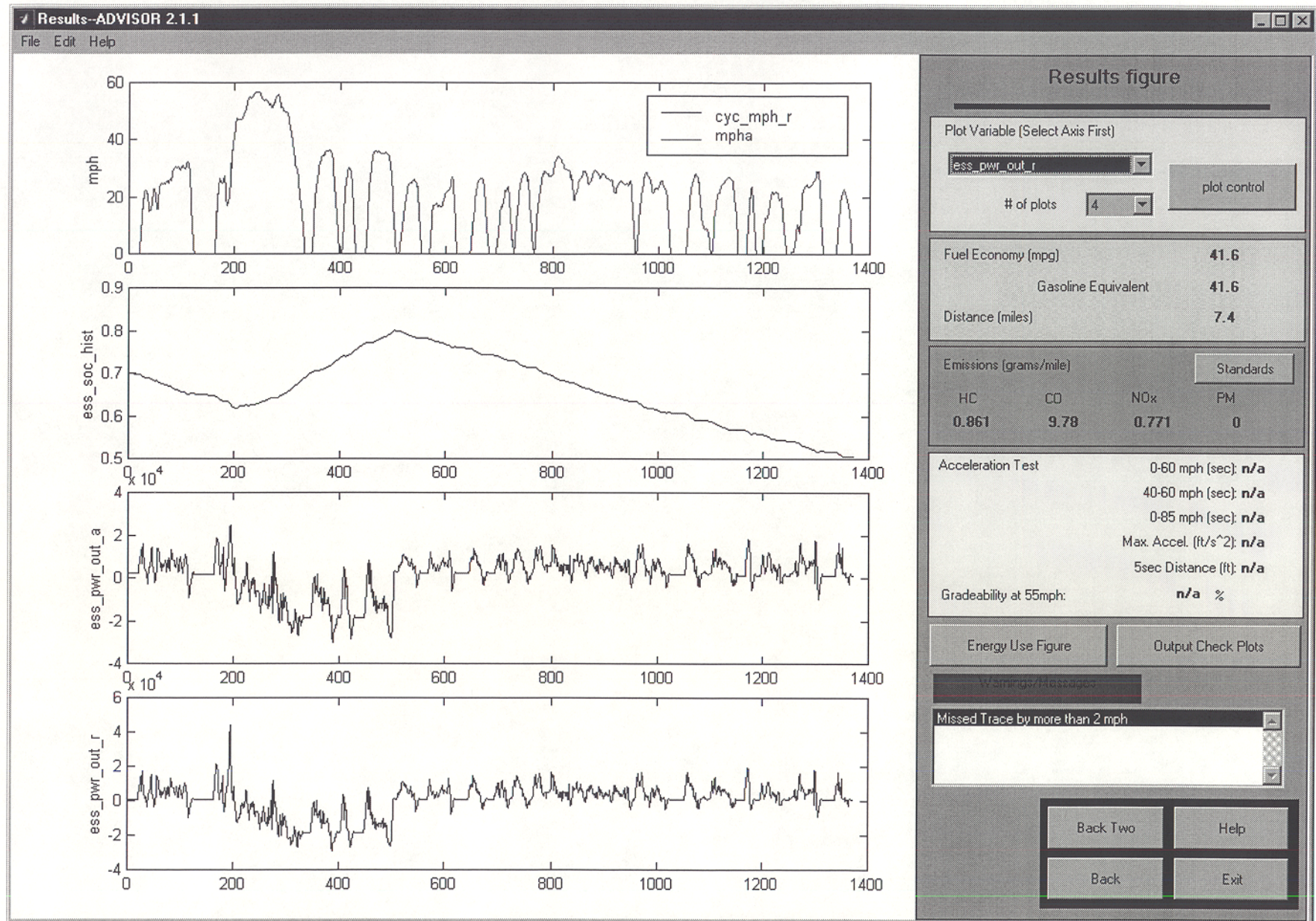
We started with dynamic neural network models trained with Optima data. The best dynamic neural network had one dynamic element and its performance (on current prediction) was:

- Prediction Error – 1.47%
- MSE – 0.12
- RSE – 0.96

Static neural networks showed better performance. The performance of the best static neural network (on current prediction) was:

- Prediction Error – 0.53%
- MSE – 0.08
- RSE – 0.99

Figure 27: Successful ANN Integration in ADVISOR



By application of the Smart-Select technique for selection of training data, the performance was further improved. The performance on current prediction was now:

- Prediction Error – 0.6%
- MSE – 0.0057
- RSE – 0.9993

The static neural network developed with Hawker data had the performance figures:

- Prediction Error – 1.16%
- MSE – 0.0247
- RSE – 0.9652

Although this was not the best performance, no improvement in performance could be achieved without improving the quality and consistency of the experimental data. The neural network's performance was comparable to that of the original, circuit-law analysis based algorithm, which was:

- Prediction Error – 0.99%
- MSE – 0.02
- RSE – 0.973

This level of accuracy proved to be adequate for the ESS simulation in ADVISOR.

8.0 Potential Areas for Investigation

- Integration of complete ANN capability into ADVISOR.
- Modeling the IC engine of the HEV.
- Optimization of IC engine performance by ANN-RSM.
- Diagnosis of IC engine problems with an ANN monitor.
- Modeling other HEV components.

9.0 Acknowledgments

We are grateful to the U.S. Department of Energy and the National Energy Renewable Lab (NREL) for sponsorship of this investigation. We are thankful to the University of Colorado, under whose auspices this research was carried out. The authors gratefully acknowledge the contribution of Dr. Yuan Li, Matthew Cuddy and David Rausen to this project.

Table I:*Description of OPTIMA data files.*

File Name	Cycle Type	Duration (sec)	Size (# Data Points)
25cc1	Charge	2378.26	80
25cc2	Charge	1625.43	55
25cd1	Discharge	2257.69	76
25cd2	Discharge	1926.19	65
40cc1	Charge	2669.47	226
40cc2	Charge	1570.37	117
40cc5	Charge	582.52	59
40cd1	Discharge	2671.09	226
40cd2	Discharge	1519.17	112
40cd5	Discharge	757.41	74
fuds10	Driving	1581.69	7404
fuds13a	Driving	1773.46	6105
fuds13b	Driving	1502.7	6049
gdstc	Driving	3333.5	5179
gdstd	Driving	5291.25	7517
<i>Total Data Points in Driving Cycles -</i>			32254

Table II:*Range of the input and output variables for the driving cycles.*


CYCLE	Power (KW)		Ampere-Hours		Voltage (Volts)		Current (Amps)	
	Min	Max	Min	Max	Min	Max	Min	Max
Fuds10	-1.59	1.69	-6.46	-5.46	10.3	16.4	-154.8	103.2
Fuds13a	-2.10	2.01	-5.39	-4.68	10.1	16.2	-207.7	124.3
Fuds13b	-2.04	1.80	-6.86	-5.94	10.1	15.2	-200.2	119.7
Gdstc	-0.80	1.99	-12.69	-1.43	10.7	16	-74.2	131
Gdstd	-1.58	1.21	-12.73	0.00	9.9	15.3	-155.4	79.2
								
Training Data	-2.106	1.797	-12.73	-0.005	9.9	16.4	-207.7	119

Table III:

Range of the variable 'Power Requested' for ADVISOR driving cycles.

	<i>Power Requested (KW)</i>	
<i>CYCLE</i>	Min	Max
ARB02	-2.4017	3.112359
HL07	-1.52347	3.292739
US06	-1.82609	2.756176

APPENDIX A: The original empirical-analytical implementation of the ESS module in ADVISOR.

In ADVISOR, the overall schematic implementation of the ESS module is as shown in Figure 5. This block accepts a power request (P_r), and depending on the state-of-charge (SOC) of the battery, returns the bus voltage (V) and current (I). The ESS output power is the product of the bus voltage and current. In the original version of ADVISOR, the algorithm was implemented as represented in Figure 6. The open-circuit voltage of the battery-pack (V_{oc}) and its internal resistance (R_{int}) are computed as functions of the SOC. Then, applying circuit law analysis, I and V are computed from V_{oc} , R_{int} and P_r . Implementation details of the ESS simulation in the MATLAB-SIMULINK graphical programming environment are shown in Figure 7. The Figure 7 comprises four internal modules:

- a module to compute pack voltage and internal resistance;
- a module to limit output power;
- a module to compute bus voltage and current;
- a module to update the battery SOC.

Block 1: Computation of Pack Open Circuit Voltage and Internal Resistance.

This block calculates V_{oc} and R_{int} given the SOC and P_r .

- (1) Interpolated look-up tables for V_{oc} and R_{int} (charging and discharging) are used to determine these parameters from the SOC for a single battery.
- (2) The appropriate resistance is chosen, depending on whether the power requirement is for charging (negative power by convention) or discharging (positive power by convention).
- (3) V_{oc} and R_{int} are scaled by the number of batteries in the pack.

Block 2: Limit Power.

This block prevents the power that is used to compute the bus current from exceeding limits imposed by three factors: SOC, equivalent circuit parameters, and the motor controller's minimum allowable voltage.

-
- (1) If an attempt is made to draw power from a depleted battery pack, the power request is limited to zero.
- (2) If $V_{oc} / 2$ is greater than the minimum motor controller voltage, then the maximum power that the battery pack can deliver would not bring the bus voltage down below the motor controller minimum voltage. In this case, the battery pack is able to produce the full power of which it is capable, $V_{oc}^2 / 4R$, and that value is used as the maximum power limit.
- (3) If $V_{oc} / 2$ is less than the minimum motor controller voltage, then the power is limited by the minimum motor controller bus voltage. In this case, the motor controller will limit power before the battery pack reaches its maximum, and so the maximum power limit reflects the effect of the minimum motor controller voltage limit.
- (4) This is where the maximum power limit is calculated, according to the formula:

$$P_r = V_{bus} \times I_{bus} = V_{bus} \times \frac{V_{oc} - V_{bus}}{R_{int}}$$

where V_{bus} is either $V_{oc} / 2$ or the minimum motor controller voltage, whichever is larger.

Block 3: Compute Current and Voltage.

I and V are computed from V_{oc} , R_{int} and P_r , applying straightforward circuit law analysis.

- (1) Kirchoff's voltage law (KVL) requires that: $V = V_{oc} - (R_{int} \times I)$.
- (2) Power is: $P_r = V \times I$. Therefore $V = P_r / I$. Combining this equation with the KVL equation of (1) yields: $(P_r / I) = V_{oc} - (R_{int} \times I)$. Multiplying both sides of the equation by I yields $P_r = (V_{oc} \times I) - (R_{int} \times I^2)$. This is the equation that is solved for I in the block diagram:

$$(R_{int} \times I^2) - (V_{oc} \times I) + P_r = 0$$

There are actually two solutions for this equation, but the larger solution is ignored as it would require larger current, and thus a lower terminal voltage, to produce the same power. All solutions that require a terminal (or bus) voltage less than half the battery pack's open circuit voltage are thus not considered.

Block 4: SOC Algorithm.

The SOC algorithm in ADVISOR is responsible for updating the SOC of the battery pack as it is subject to charging or discharging during service. The procedure is described as a series of steps.

Discharging -

- (1) The average discharge current is computed as the sum of the net charge that has been withdrawn from the battery pack, divided by the total duration of discharge period. By convention, discharge current is positive.
- (2) The *Peukert Equation* is applied to compute the effective maximum charge capacity (in units of Ampere-Hours) of the battery pack corresponding to the average discharge current.
- (3) The effective maximum charge capacity is used to determine the effective starting charge, in Ampere-Hours, by multiplying it with the initial SOC of the battery pack.
- (4) The total charge withdrawn from the battery pack (in units of Ampere-Hours) is computed as the product of the average discharge current and duration of the discharge period. This quantity is subtracted from the effective starting charge as obtained by applying the Peukert equation in (3). This yields the remaining charge of the battery pack in Ampere-Hours.
- (5) The updated SOC is the ratio of the battery pack's remaining charge as computed in (4) to the effective maximum charge capacity derived from (3).

Charging:

The procedure is the same as for discharging except that the charging current (negative by convention) is scaled by a *coulombic efficiency factor*.

APPENDIX B: The C++ program for Smart-Select.

This section briefly presents the C++ program that implements the Smart-Select algorithm with a brief description.

Smart-Select is applied to the input file `IN_FILE` and the file `OUT_FILE` is generated as output. Both files are ASCII text files, and contain data in column format. The first column of the input file is the ‘key’, with reference to which the program generates the output. The other columns contain the variable data. The way the key is used is as follows. From consecutive rows sharing the same key, only the first row is selected, stripped of the key, and written to the output file. The key may be generated easily in MS-EXCEL spreadsheet software, by reference to the time data, for example.

```
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>

#define IN_FILE "pre_file.txt"
#define OUT_FILE "post_file.txt"

int main()
{
    double var1, var2, var3, var4, var5, var6, var7, var8, var9;
    char dummy;

    ifstream fin;
    ofstream fout;

    fin.open(IN_FILE);
    fout.open(OUT_FILE);

    if (fin.fail())
    {
        cout << "Failed to open in read mode - file " << IN_FILE;
        cin >> dummy;
        exit(1);
    }
}
```

```

    if(fout.fail())
    {
        cout << "Failed to open in write mode - file " << OUT_FILE;
        cin >> dummy;
        exit(1);
    }

    fin >> var1 >> var2 >> var3 >> var4 >> var5 >> var6 >> var7
        >> var8 >> var9;
    double temp = var1;
    fout << var1 << " " << var2 << " " << var3 << " "
        << var4 << " " << var5 << " " << var6 << " "
        << var7 << " " << var8 << " " << var9 << endl;

    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
    cout << "1st - " << var1 << " 2nd - " << var2 << " 3rd - "
        << var3 << " 4th - " << var4 << " 5th - "
        << var5 << " 6th - " << var6 << " 7th - "
        << var7 << " 8th - " << var8 << " 9th - "
        << var9 << endl;

    while (fin >> var1 >> var2 >> var3 >> var4 >> var5 >> var6
        >> var7 >> var8 >> var9)
    {
        if (var1 != temp)
        {
            temp = var1;
            fout << var1 << " " << var2 << " " << var3
                << " " << var4 << " " << var5 << " "
                << var6 << " " << var7 << " "
                << var8 << " " << var9 << endl;

            cout.setf(ios::fixed);
            cout.setf(ios::showpoint);
            cout.precision(2);
        }
    }

    fin.close();
    fout.close();

    cout << "\nEND OF PROGRAM.\n";
    cin >> dummy;

    return 0;
}

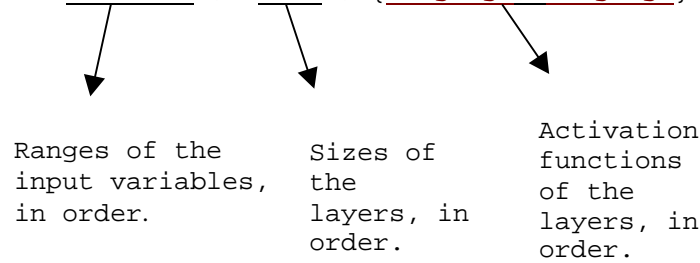
```

APPENDIX C: Porting between CUANN and MATLAB .

For this project, the CU-ANN[®] was required to be ported into the MATLAB[®] environment of ADVISOR. The porting is explained here with the help of a simple example.

To port a neural network developed by the CUANN[®] software into the MATLAB[®] environment, it is first required to define a neural network of identical structure in MATLAB[®], with its Neural Network Toolbox. To define a feed-forward neural network in MATLAB[®], the Neural Network Toolbox provides the command `newff`. The anatomy of this command is described with the following example -

```
net = newff([0 1; 0 1], [3, 2], {'logsig', 'logsig'});
```



In this example, the variable `net` is a two-layer, feed-forward neural network. That means there is one hidden layer, the other layer being the output layer. The hidden layer has three neurons and there are two output neurons. The activation function of both layers is a logistic sigmoid. Note that the range of each input variable is specified as `[0 1]`, since the CU-ANN[®] automatic pre-processor normalizes the input variables in that range.

The next step is to load the weights of the neural network from the CU-ANN[®]. For this purpose, text files are created from the CU-ANN[®] as follows:

- a text file for the bias weights of each layer.
- a text file for the interconnection weights of each layer.

According to the format of the weight files in the CU-ANN[®], the weights are organized layer-wise, as matrices. The bias weights of a layer and the weights of interconnections with its previous layer are organized as a matrix. The weights of an individual neuron are organized in a row, with the first element in a row being its bias weight. The rows are

arranged in the order of the neurons. In the example cited, the weights of the hidden layer would be represented as follows -

	Bias	Input 1	Input 2	Input 3
Neuron 1
Neuron 2
Neuron 3

The text file for loading the bias weights of a layer would comprise the first column of its weight matrix in the CU-ANN weight file. The text file for loading its interconnections with its previous layer would comprise the remaining columns. In the example cited, the weight assignment would require four text files, two for each layer. The weight assignment in MATLAB[®] would be as follows;

```
net.IW{1, 1}=weights_10;
net.b{1, 1}=weights_10b;
net.LW{2, 1}=weights_21;
net.b{2, 1}=weights_21b;
```

In the MATLAB[®] environment, the hidden layer connecting to the input layer is identified as `net.IW{1, 1}`. The weights of any other layer are identified as `net.LW{a, b}` where `a` is the number of the layer and `b` is the number of the layer with which it is interconnected. The numbering system starts with the first hidden layer in the feed-forward direction, which is standard practice. The same notation has been used in this example to label the text files, with a trailing 'b' indicating that the file contains bias weights.

Now, the neural network is ready for prediction. The Neural Network Toolbox provides the command `sim` for this purpose. The anatomy of this command is as follows:

```
a = sim(net, in)
```

```

graph TD
    A["a = sim(net, in)"]
    A --> B["output:  
vectors  
in row  
format"]
    A --> C["network  
variable"]
    A --> D["input:  
vectors in  
row format"]

```

Note that the input vectors are presented to the network in row format. Correspondingly, the outputs are also in row format. The inputs require to be normalized in the range [0, 1]. The normalization is performed as follows:

$$in^i(normalized) = \frac{in^i - in_{\min}^i}{in_{\max}^i - in_{\min}^i}$$

where the superscript labels the input variable, and the maximum and minimum values refer to the training data. The CU-ANN normalizes the output variables in the range [0.2, 0.8], so the output a has to be de-normalized as follows:

$$a^j(de-normalized) = \frac{a^j - 0.2}{0.8 - 0.2} \times (a_{\max}^j - a_{\min}^j)$$

where, the superscript labels the output variable, and the maximum and minimum values refer to the training data.

This concludes the discussion on portability of the CU-ANN[®].

REPORT DOCUMENTATION PAGE			Form Approved OMB NO. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE August 1999	3. REPORT TYPE AND DATES COVERED Technical report		
4. TITLE AND SUBTITLE Neural Network Based Energy Storage System Modeling for Hybrid Electric Vehicles			5. FUNDING NUMBERS HV906010	
6. AUTHOR(S) S.R. Bhatikar, R.L. Mahajan, K. Wipke, and V. Johnson				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) National Renewable Energy Laboratory 1617 Cole Blvd. Golden, CO 80401-3393			8. PERFORMING ORGANIZATION REPORT NUMBER TP-540-26934	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Department of Energy 1000 Independence Avenue Washington, D.C. 20585			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT National Technical Information Service U.S. Department of Commerce 5285 Port Royal Road Springfield, VA 22161			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The modeling of the energy storage system (ESS) of a hybrid electric vehicle (HEV) poses a considerable challenge. The problem is not amenable to physical modeling without simplifying assumptions that compromise the accuracy of such models. An alternative is to build conventional empirical models. Such models, however, are time-consuming to build, as well as data intensive. In this paper, we demonstrate the application of an artificial neural network to modeling the ESS. We show that neural network models can accurately capture the complex, non-linear correlations accurately. Finally, we propose and deploy our new technique, Smart Select, for designing neural network training data.				
14. SUBJECT TERMS			15. NUMBER OF PAGES 43	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT L	

